An Investigation of Bluetooth Mergence with Ultra Wideband

van der Linde, E.
Department of Electrical, Electronic and Computer Engineering
University of Pretoria, South Africa

Building 16, 1-119
CSIR Campus
Meiring-Naudé Road, Brummeria
Pretoria, South Africa

etienne.vdlinde@xstreamflow.com

Hancke, G.P.
Department of Electrical, Electronic and Computer Engineering
University of Pretoria

Engineering 1 Building, 15-5
Cnr Lynnwood and University Roads
University of Pretoria
Pretoria, South Africa

Gerhard.Hancke@up.ac.za

Abstract

This article investigates the viability of a merger between Bluetooth and Ultra Wideband by developing and implementing a Bluetooth over Ultra Wideband data channeling system with an emphasis on the design of an economically feasible system and the viability of the proposed merger points. The designed system is then applied to transfer Bluetooth data over a compliant UWB Medium Access Control (MAC) and Physical (PHY) layer radio channel to prove its functioning and unveil the advantages UWB may provide to Bluetooth with regards to speed and distance.

Keywords

Bluetooth; Ultra Wideband; ECMA-368; ECMA-368; UWB; protocol; BToUWB; WPAN; AMP; performance; bandwidth; ACL; SCO; RFCOMM

1. Introduction

In order to retain its market success as a dominant short-range low-power wireless technology, Bluetooth requires keeping up with technological developments and meeting the bandwidth and power demands of today's high-tech wireless peripherals. Although Ultra Wideband and Bluetooth have been compared with each other on a number of levels, the literature study revealed that little integration efforts between UWB and Bluetooth have been published yet, despite the Bluetooth Special Interest Group's (SIG) announcement to incorporate UWB as its high-speed wireless alternative radio technology. The research performed focuses on fusing these two wireless technologies on a feasible level with economically available hardware and open-source software, and explore the potential that such a merger will disclose. Viably integrating these protocols may combine the advantages of Bluetooth as a well established widely implemented technology with that offered by UWB such as speed and distance. Ultimately it may provide upper layer based Bluetooth devices with the bridging functionality to exploit the advantages offered by Ultra Wideband.

Ultra Wideband (UWB), a short-range radio technology spreading communication information over a large spectrum with low energy requirements, has been proposed to fulfill this need, and in 2007 the Bluetooth Special Interest Group (SIG) announced its intensions to incorporate Ultra Wideband as ad-on high-speed radio technology in the Bluetooth 4 release due 2010/11. As a relatively new player, UWB needs to align itself with established technologies in order to gain momentum, prompting the WiMedia Alliance in March 2009 to announce its intention to disband and transfer all current and future specifications to the Bluetooth SIG, Wireless UWB Promoter Group and USB Implementers Forum.

The Bluetooth 3.0 specification released in April 2009 incorporates an Alternative MAC/PHY (AMP) feature, which enables the use of an alternative high-speed radio for transporting Bluetooth profile data, whilst the Bluetooth radio is still used for device discovery and connection setup. Currently only supporting 802.11 (Wi-Fi), providing Bluetooth with increased transfer rates of 24 Mbps over longer distances, this new AMP feature paves the way for incorporation of the low-power high-bandwidth UWB radio technology. The findings of this paper may aid in these proposed efforts.

The advantages offered by UWB combined with the inherent research and development efforts made UWB technology fairly inaccessible to researchers with low budgets. This article aims to provide a framework for setting up a cost-effective Bluetooth over UWB (BToUWB) system merged at a software level HCI interface by use of open-source operating systems and protocol stacks. The system can be implemented to transmit generated Bluetooth traffic over a compliant UWB link for further in-depth analyses of such a proposed merger.

2. Protocol Architectures

An overview of the Bluetooth and Ultra Wideband protocol stacks highlighting details pertaining to our merger analyses is briefly provided in this section. More detailed information on the two protocols may be obtained from the Bluetooth [1] and ECMA [2] [3] specifications.

2.1. Bluetooth

Bluetooth [1] aimed to be the single digital wireless protocol for connecting multiple devices over short range creating wireless personal area networks (PANs). It was primarily designed for low power consumption over short distances. Figure 1 shows the layered Bluetooth protocol stack, with each layer aiming to provide transparent functionality to its upper layer.

A Bluetooth radio module operates in the unlicensed 2.4 GHz Industrial, Scientific, and Medical (ISM) band, and avoids interference from other signals by using a spectrum spreading technique of frequency hopping between 79 available transmission channels. When two or more Bluetooth devices establish a connection, one device will take the role of master whilst the others will be slaves, forming a piconet.

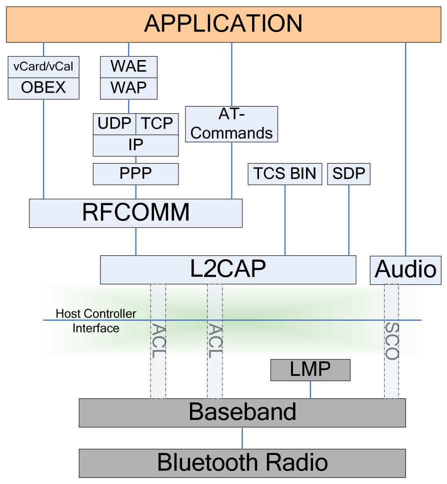

Figure 1. The Bluetooth protocol stack [1]

The baseband handles two types of links: Synchronous Connection Oriented (SCO) and Asynchronous Connection-Less (ACL), compared in Table 1.

Table 1. Bluetooth Baseband Physical Links

Attribute	SCO	ACL
Link type	Symmetric point-to-point	Point-to-multipoint

Slot usage	reserved	Per-slot basis
Timing	Synchronous	Asynchronous
Data type	Voice (64 kB/s speech)	Data packets
Analogy	Circuit switched	Packet switched
Error correction	No retransmission	Packet retransmission
Links allowed	Up to 3	1

The Host Controller Interface (HCI) provides a command interface to the baseband Link Controller (LC) and Link Manager (LM) and access to the hardware status and control registers. The HCI exists across the following three sections with a functional part in each:

1. *HCI driver* - receives and parses asynchronous events
2. *HCI firmware* - implements hardware commands
3. *HCI Transport Layer* - layers by which the driver communicates with the host system

The Logical Link Control and Adaptation Protocol (L2CAP) provides connection-oriented and connectionless data services to various higher protocol levels whilst multiplexing between them. L2CAP only supports best effort ACL 'channels' with data packets up to 64 kilobytes in length.

The RFCOMM layer provides the emulation of (RS232) serial ports over L2CAP links. Up to 60 simultaneous connections over a single session is supported, whilst providing various flow control methods.

A tailored Service Discovery Protocol (SDP) enables an application to discover which services are provided by a linked Bluetooth device by either searching or browsing its service records. The Bluetooth specification also provides a number of defined profiles that describe how the different parts of the protocol stack can be applied to fulfill a desired function for a Bluetooth device.

2.2. Ultra Wideband

Ultra Wideband is a relatively new radio technology designed for wireless data transfers of high bandwidth and low power requirements. Ultra Wideband was traditionally accepted as pulse radio, but the Federal Communications Commission (FCC) now defines UWB in terms of a transmission from an antenna for which the emitted signal bandwidth exceeds the lesser of 500 MHz or 20% of the center frequency. The signal spreading can be achieved by using either a pulse-based system which instantaneously occupies the complete allowed bandwidth, or an aggregation of narrow band carriers, for example the Multi-Band Orthogonal Frequency Division Modulation (MB-OFDM) scheme promoted by the WiMedia Alliance[1].

The ECMA-368 [2] standard, from hereon only denoted as UWB, specifies the popular WiMedia Ultra Wideband physical layer (PHY) and medium access control (MAC) sub-layer for a high-speed short range wireless network, supporting data rates of up to 480 Mb/s. The standard only defines a PHY and MAC layer to act as transmission sub-layers for various different higher level protocols, like Wireless USB, Internet Protocol (IP) in the form of WiNet, and other application like Bluetooth, as depicted in Figure 2.

[1] WiMedia Alliance is a non-profit organization supported by more than 200 member corporations and research institutions, http://www.wimedia.org

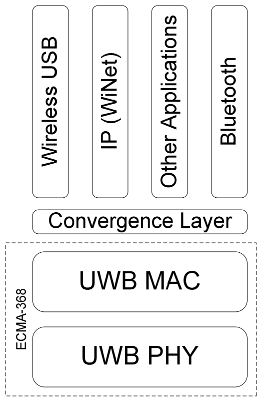

Figure 2. ECMA-368 UWB Protocol Stack

The standard divides the unlicensed 3.1GHz – 10.6 GHz frequency spectrum into 14 bands of 528 MHz, grouped into four band groups of 3 each and a fifth band group of 2 bands. Frequency domain spreading, time-domain spreading, and forward error correction (FEC) with rates 1/3, 1/2, 5/8 and 3/4 are used to vary the data rates between 53.3 Mb/s, 80 Mb/s, 106.7 Mb/s, 160 Mb/s, 200 Mb/s, 320 Mb/s, 400 Mb/s and 480 Mb/s. The coded data is then spread over the frequency by either interleaving it over three bands with Time Frequency Interleaving (TFI) or transmitting it over a single band using Fixed Frequency Interleaving (FFI).

The ECMA-369 [3] standard specifies the interface between implementations of the PHY and MAC as specified in ECMA-368. Table 2 shows the MAC sub-layer prioritized medium access schemes used for data transfer, which together with other policies ensures equitable access to the bandwidth.

Table 2. MAC Prioritized Medium Access Schemes

Traffic type	Medium Access	Protocol
Isochronous	Time Division Multiple Access (TDMA)	Distributed Reservation Protocol (DRP)
Asynchronous	Carrier Sense Multiple Access (CSMA)	Prioritised Contention Access (PCA)

The MAC protocol is specified with respect to an individual device and its neighborhood forming a logical ‘beacon’ group to facilitate contention free frame exchange. When a device is enabled, one or more channels are scanned for a beacon. Once detected, the device synchronizes its beacon period (BP) with it and uses this channel for exchanging data with members in its beacon group. Neighboring devices exchange data frames on a TDMA basis by use of a timing structure called a superframe of 65 536 μs, which is in turn composed of 256 medium access slots (MASs) of 256 μs each.

Information discovery is supported by broadcasting Information Elements (IEs) in beacon frames or requesting it in probe commands.

2.3. Mergence

There exist some similarities between UWB and Bluetooth as wireless communication protocols. Both operate over a short range (1 – 10 m) with minimal power requirements. Although the two standards make use of different unlicensed frequency bands, it has been shown that under extreme interference conditions UWB devices may have an impact on Bluetooth networks [4]. Both UWB and BT protocols have a layered architecture. This clustering of functionality eased our aim of imposing a clean break within the BT stack to merge with the lower-level UWB stack. The HCI layer in Bluetooth provides a clear separation between upper layer BT protocol stacks, commonly implemented in software, and lower device dependant layers, commonly implemented in hardware. The interface consists of commands and events which can easily be imitated by a UWB convergence layer [5].

A whole number of differences between the two protocol stacks need also be considered. Unlike Bluetooth’s master-slave scheme, the UWB MAC services are fully distributed with all devices providing the required functions and no device acting as central coordinator, enabling UWB devices to communicate directly with each other, and not only react to a single master device. Inherently there is also a difference in the device discovery and connection setup process applied by the two standards. Whilst Bluetooth make use of page scanning, UWB uses beacons in a beacon period. In Bluetooth's piconet with single master scheme, TDMA based channel access is also handled differently than with UWB's prioritized medium access schemes based on TDMA and CSMA. With Bluetooth's multiple piconet network topology all devices in range are synchronized and visible and don't require explicit scanning to be discovered.

3. Methodology

The scope of this project is restricted to conditions where UWB is applied as physical transport layer for upper layer Bluetooth data transmissions, and not other Bluetooth housekeeping tasks like device discovery and connection setup. For these purposes, prior to channeling data over a UWB connection, a compliant Bluetooth and UWB connection were configured between two Linux [6] enabled computers by use of Bluetooth and UWB enabled USB dongles. On the software level the official Bluetooth stack for Linux called BlueZ [7] were used to implement a modified Bluetooth system. BlueZ has been included in the Linux kernel since version 2.4.6., and supports all core Bluetooth functionalities between the application and baseband layers shown in Figure 3.

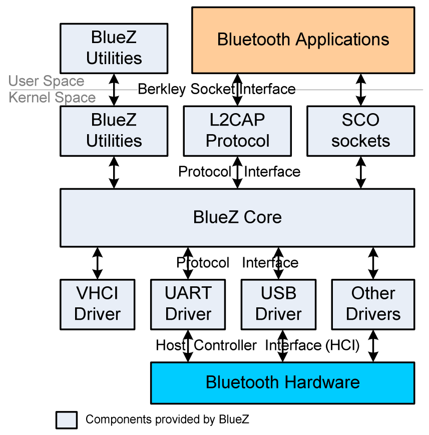

Figure 3. BlueZ Linux Bluetooth protocol stack [8]

Despite various efforts at the time of starting this project, affordable Ultra Wideband hardware could not be obtained due to manufacturers' lack of research programs and primary concern with high volume commercial prospects. This stagnation led to an implementation of the proposed system over a simulated UWB link [9]. In 2008 UWB enabled hardware finally became available in the form of Wireless USB dongles [10], containing the Intel Wireless UWB Link 1480 Ultra Wideband MAC [11]. By modifying the developmental driver for this hardware [12] and loading the Wireless USB dongles with WiMedia Logical Link Control Protocol (WLP) specific Intel firmware, a UWB link were established under the Linux environment. This Linux UWB driver supported UWB scanning, beacon groups, WLP and more.

To emulate the designed communication system, two computers configured with a BlueZ enabled version of Linux compatible with the Linux UWB driver, first needs to be set up with a functioning Bluetooth link by making use of the BlueZ utilities like `HCICONFIG` and `L2PING`. Simultaneously, a functioning UWB communication channel needs to be configured between the two computers by use of UWB hardware compatible with the Linux UWB driver. The Linux kernels needs to be patched with the driver, after which the `scan` and `beacon` commands can be used to set up a UWB link. A WLP network should also be configured on top of the UWB driver interface [13]. Figure 4 gives an overview of the implemented system design.

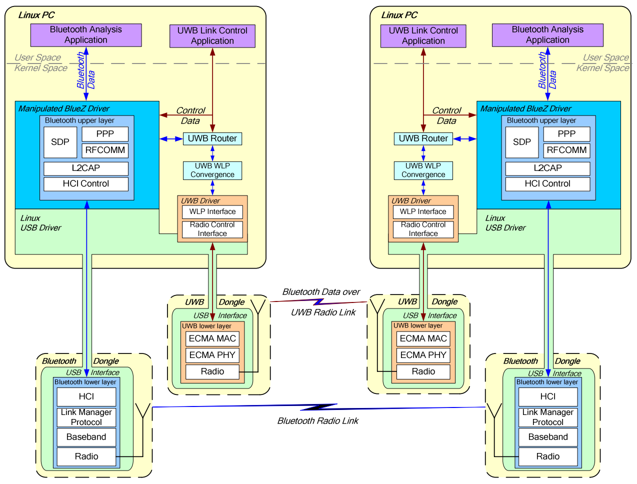

Figure 4. Implemented system diagram

In order to channel the Bluetooth link's data over the UWB link, two kernel-space modules called `UWB router` and `WLP convergence` were developed to act as gateway between the Bluetooth and UWB software stacks, modifying the data packets for compliance to each protocol. Following is a detailed explanation of how these modules integrate the two software stacks, whilst Figure 5 provides a graphical interpretation of the routing functions performed.

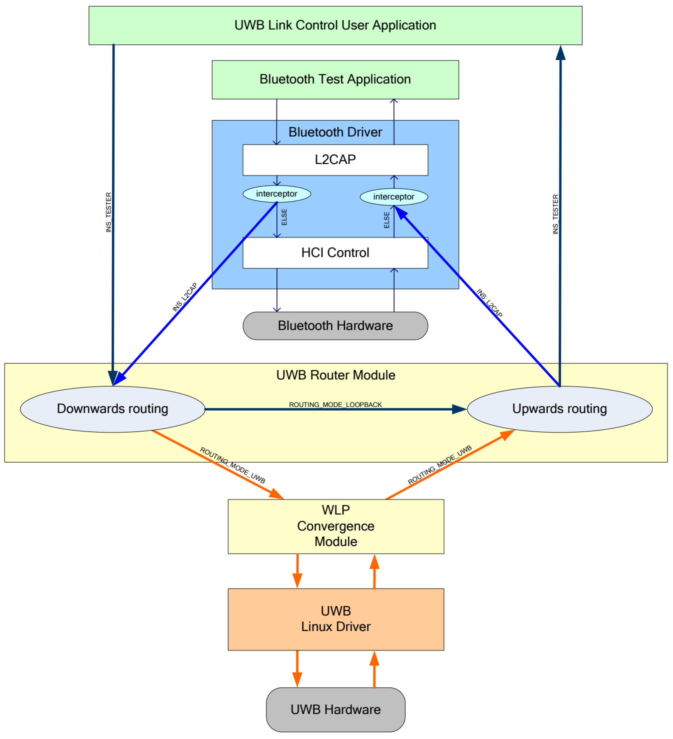

Figure 5. UWB and Bluetooth software integration

The Linux BlueZ files needed to be modified for capturing the relevant HCI command packets sent to the HCI sub-layer by upper layer L2CAP and SCO implementations, and routing these packets to the `UWB router` module. More specifically, calls made to the `hci_send_acl()` function in the `l2cap.c` file and `hci_send_sco()` function in the `sco.c` file were targeted. Similarly the module needs to emulate lower level HCI events to the Bluetooth L2CAP and SCO upper layers upon receipt of packets from the UWB subsystem. To propagate data back into the Bluetooth system, a new HCI connection is created upon receipt of an event according to the header data followed by calls to either the `l2cap_recv_acldata()` or `sco_recv_scodata()` functions.

The `WLP convergence` module opens a server socket over WLP to the local UWB hardware device, opens a client socket over WLP to the target UWB device and consequently listens for incoming connections from the remote UWB hardware. The `WLP convergence` module then encapsulates and passes data received from the `UWB router` module on to the Linux UWB driver via the WLP connection, from where the UWB Radio Controller (RC) propagates it through the UWB link from computer A to B. Data arriving at computer B is propagated back up to the `UWB convergence layer` through implemented call-back functions, from where it is injected into the remote Bluetooth subsystem according to the embedded header information via the `UWB router` module.

To generate valid Bluetooth data for performing the desired tests, a `Bluetooth test application` has been developed according to proposed Bluetooth user level methods [14]. This program accesses the Linux Bluetooth subsystem via socket connections for sending and receiving either asynchronous ACL data over the L2CAP layer or isochronous SCO streaming audio data over BlueZ's SCO implementation. Synchronous SCO data are transmitted via the high speed asynchronous UWB link and buffered on the receiver side, to be timely reinserted into the synchronous receiver stream.

A `UWB link control` application was also developed to interface to the `UWB router` module for enabling and disabling packet routing over the UWB subsystem during runtime. A more detailed description of the developed software modules is available in the project report [15].

4. Results

This section lists the results obtained from analyzing ACL and SCO data sent across the Bluetooth-over-UWB system implementation as described in section 3. It serves to prove the functionality of the proposed system design, including merger points between the two technologies. The actual obtained results together with discrepancies from expected performance are also explained.

4.1. Bluetooth Asynchronous Data Analysis

The ACL data generated and transmitted by the user level application were remotely bounced with the turnaround time logged for comparison over various transmission distances. Figure 6 through Figure 8 shows a comparison between the measured asynchronous data throughput over a Bluetooth only link using DH3 packets against that obtained from the implemented BToUWB system link for various transmission distances. A developed Bluetooth data generator program *btan* were used to send and loopback data over the two wireless links.

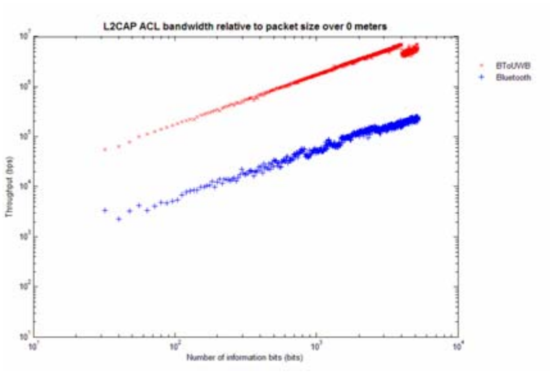

Figure 6. BToUWB ACL data throughput (0m)

Figure 6 shows how the higher bandwidth offered by the UWB subsystem is reflected in the BToUWB 50 fold throughput increase over the traditional Bluetooth bandwidth for various packet sizes at 0 meter transmission distance. A positive sloping graph depicts how the bandwidth is more optimally utilized with increasing packet sizes: a larger proportion of each packet contains actual payload data versus overhead included.

At 2 meters, showed in Figure 7, similar bandwidth improvements can be observed for the BToUWB system, although the detrimental effect of the increased transmission distance on the Bluetooth link is starting to prevail. An overall bandwidth increase compared to Figure 6 can also be observed, attributed to the improved utilization of the channel by actual payload data versus overhead for larger packets.

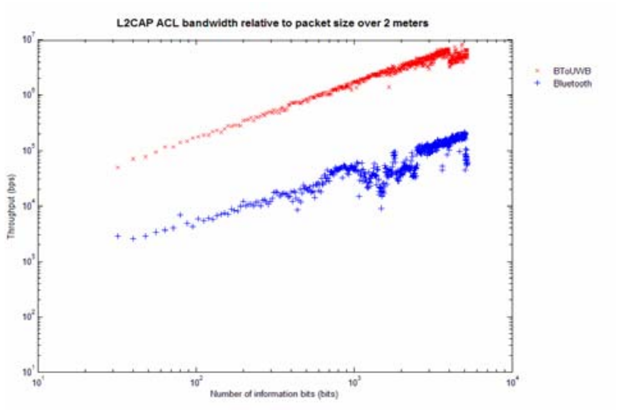

Figure 7. BToUWB ACL data throughput (2m)

A discontinuity in the BToUWB L2CAP data can be observed around 3500 information bits in Figure 6 to Figure 7. This may be ascribed to the extra overhead kicking in at the UWB driver level for fragmenting packets to fit into 512 byte UNTD packet sizes. The 3500 / 8 ≈ 440 bytes of information, coupled with the BToUWB system overhead showed in Table 3, will result in excess of 512 byte packets requiring fragmentation.

Table 3. Packet overhead byte allocation

Level	*Defined in*	*Number of overhead bytes*
L2CAP	`l2cap.h`	4
L2CAP external routing	`l2cap.h`	16
UWB Router	`uwb_router_src.h`	4
WLP Conversion / Network	`socket.h`	16
WLP	`i1480-wlp.h`	12
Total		52

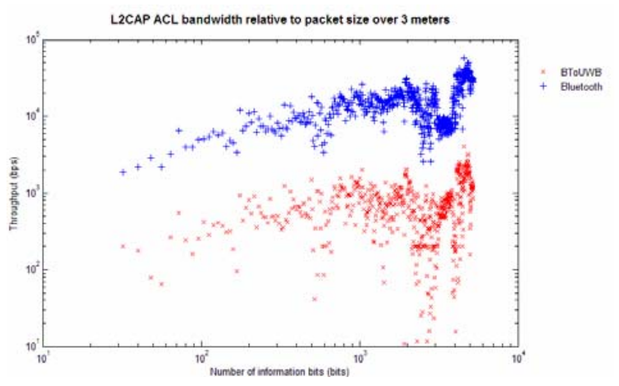

Figure 8. BToUWB ACL data throughput (3m)

Once the transmission distance was increased to 3 meters, both the BToUWB and Bluetooth channels started to decay in quality. Surprisingly the BToUWB system suffered much more impediments than the traditional Bluetooth link at this distance. This is much better evident in Figure 9, which compares the bandwidth of the Bluetooth and BToUWB links at various transmission distances.

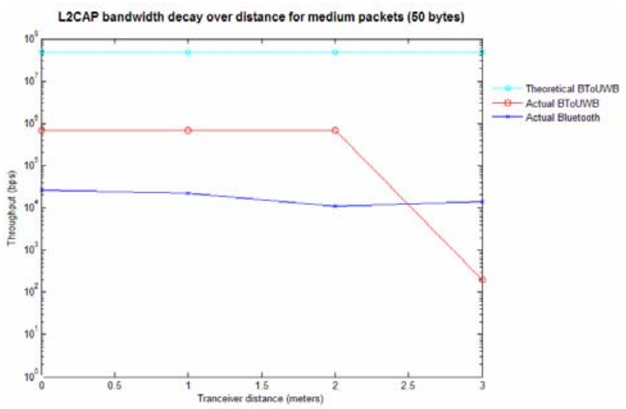

Figure 9: BToUWB ACL data throughput vs distance

The theoretical expected throughput shown for comparison in Figure 9 was obtained by assuming that the radio channel is the bottleneck in the system, whilst calculating the total channel throughput as:

$$Throughput_{Total} = \frac{Information\_Bits_{Total}}{Transmission\_Time_{Radio}} \tag{1}$$

$$\therefore Transmission\_Time_{Radio} = \frac{Information\_Bits_{Application} + Information\_Bits_{Overhead}}{Throughput_{Total}} \tag{2}$$

If we assume that the time it takes for the data to progress through the rest of the channel can be estimated as 80% of the time it takes to progress through the radio channel for high speed UWB, the actual data throughput seen by an upper layer Bluetooth application can be written as:

$$Throughput_{Application} = \frac{Information\_Bits_{Application}}{1.8 \times Transmission\_Time_{Radio}} \tag{3}$$

Substituting equation (2) into (3) we obtain the following:

$$Throughput_{Application} = \frac{Information\_Bits_{Application} \times Throughput_{Total}}{1.8 \times \left(Information\_Bits_{Application} + Information\_Bits_{Overhead}\right)} \tag{4}$$

If we assume a 200 Mbps Total Channel Throughput for the UWB radio channel, as advertised, and assume 52 bytes to be the number of overhead information bytes, as described in Table 3, resulting in 52 x 8 = 416 *Overhead Information Bits*, we can calculate the theoretical data throughput that should be seen by an upper layer Bluetooth application using the BToUWB system for each of the investigated packet sizes, as shown in Table 4. This computation ignores any other channel effects like interference, delay, path loss or fading.

Table 4. Application Throughput Calculation

Packet Size	Application Information Bits	Total Throughput (Mbps)	Overhead Information Bits	Application Throughput (Mbps)
5	40	200	416	9.75
50	400	200	416	54.47
500	4000	200	416	100.64

A reason for this dramatic performance decay above 2 meters for the system implementing a UWB communication channel, may be due to ineffective or faulty introductory hardware or the inefficient unreleased WLP firmware obtained from Intel Corporation without any guarantees.

Similar to the ACL data results obtained, Figure 10 shows how a Bluetooth RFCOMM link implementing the BToUWB subsystem experienced around 50 fold throughput increase over a short transmission distance, whilst Figure 11 portrays a dramatic decay in the same channel's quality when the transmission distance reached more than 2 meters.

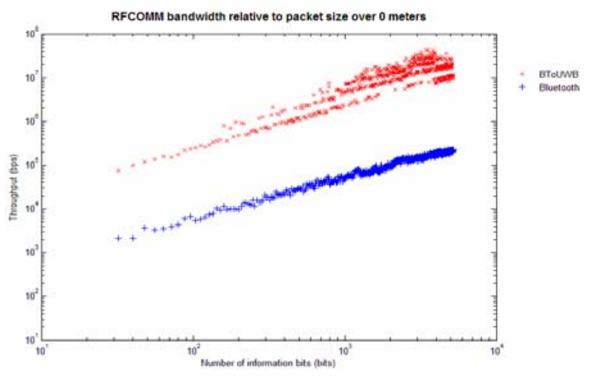

Figure 10: RFCOMM data throughput over 0 m transmission distance

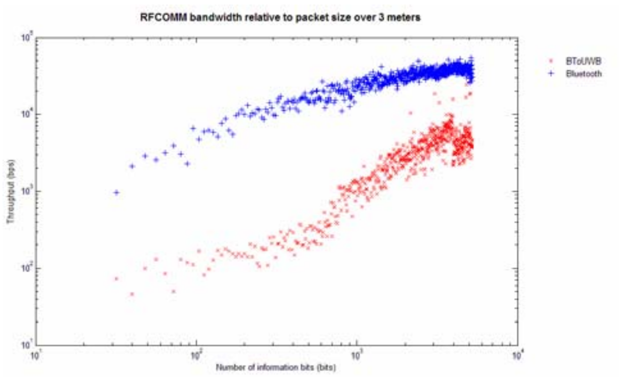

Figure 11: RFCOMM data throughput over 3 m transmission distance

Figure 12 shows a clearer comparison of the effect of transmission distance on RFCOMM bandwidth quality. This similarity to the ACL results obtained in Figure 9 is ascribed to the fact that Bluetooth RFCOMM implements ACL channels for data transfer (see Figure 1) and will therefore be subject to the same channel effects.

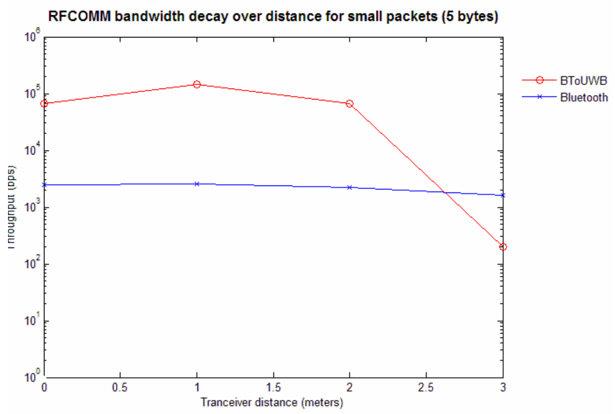

Figure 12: RFCOMM data throughput vs. transmission distance for small packets

4.2. Bluetooth Isochronous Data Analysis

For synchronous data analyses, audio data were converted to the Bluetooth compliant PCM format by use of the SoX utility. The raw audio data were then transmitted over the BToUWB system by use of the test application, and stored on the remote system for comparison with the input data, as in Figure 13.

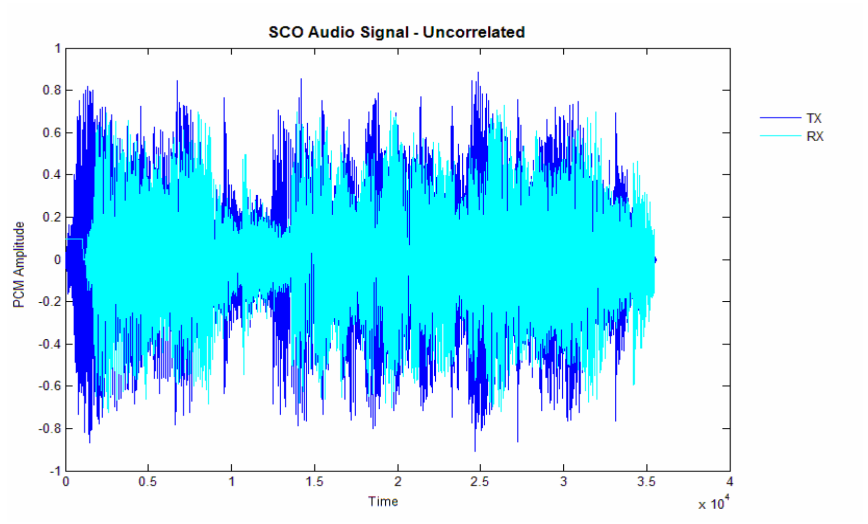

Figure 13: BToUWB transmitted vs received SCO audio data

After time-correlating the samples, the Peak Signal-to-Noise Ration (PSNR) is calculated to evaluate the quality of the received audio signal. This is achieved by first calculating the Mean Square Error (MSE) between the two signals:

$$MSE_{SCO} = \frac{1}{m}\sum_{i=0}^{m-1}\left|Tx_{SCO}(i) - Rx_{SCO}(i)\right|^2 \qquad (5)$$

Where *Tx(i)* is the transmitted audio samples, *Rx(i)* is the received audio samples, and *m* is the number of audio samples transmitted. The PSNR is then calculated by the following equation:

$$PSNR_{SCO} = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE_{SCO}}\right) \qquad (6)$$

Figure 14 compares PSNR of left and right channel audio data received over the BToUWB channel to the PSNR of the same audio received over the traditional Bluetooth channel for various distances where a low quality (8kHz stereo 16-bit audio sample rate) PCM audio signal were used as input.

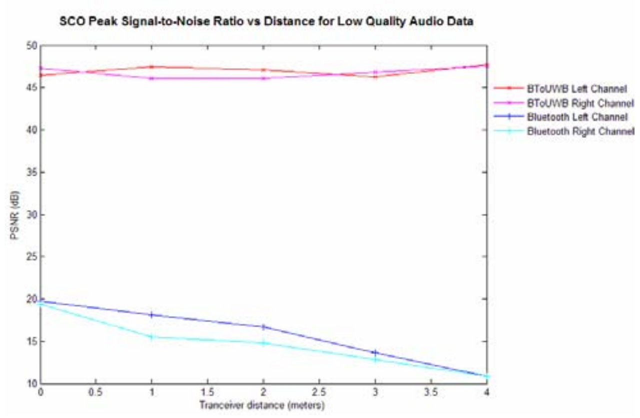

Figure 14: SCO PSNR vs transmission distance

Figure 14 shows how the implemented BToUWB system provided drastically improved SCO channel quality. Whilst the Bluetooth SCO channel quality clearly deteriorates as the transmission distance increases, the BToUWB SCO channel quality remained consistently high. From a system design viewpoint, it is in fact expected that the BToUWB SCO data should not bare any quality loss at all, whereas the measurements portrayed in Figure 14 shows a minimal loss. This can be owing to a cut-off from the tail end of the received data samples due to the timing offset between the transmitted and received data.

5. Conclusion

A BToUWB system were successfully implemented with Bluetooth and UWB hardware, showing that UWB can indeed be implemented as lower layer radio communication for Bluetooth enabled devices with easily accessible hardware and software. The system enabled the transmission and analyses of Bluetooth asynchronous L2CAP and RFCOMM data and synchronous SCO data over an Ultra Wideband link. The system also proved the HCI layer to be a reliable mergence point between the Bluetooth and UWB protocols.

The ACL and RFCOMM results obtained in section 4.1 confirms that the Bluetooth protocol stack will indeed benefit in terms of bandwidth when tunneling L2CAP ACL data over a UWB radio link instead of a Bluetooth radio link for various packet sizes and transmission distances of 0 to 2 meters. However, at a wireless transmission distances above 2 meters, the implemented UWB link quality rapidly decays against expectations to less than that of the Bluetooth link. The overall bandwidth for distances less than 2 meters is also more or less 100 times less than the theoretical expected rate. This may be due to defects in the pilot UWB radio hardware and developmental versions of Intel i1480 firmware implemented for the developed BToUWB system, and also insufficient dataflow optimization in the Linux UWB driver. At a transmission distance of 3 meters, both the Bluetooth and BToUWB links suffered severe quality impairments, especially with increased packet sizes. These results obtained then propose a selection policy for L2CAP ACL data to only be transmitted over the UWB link for transmission distances less than 2 meters, requiring employment of the position measurement functionality offered by UWB technology. The dramatic quality increase obtained for synchronous SCO data is due to the high speed channeling of synchronous data over a loss-less asynchronous UWB connection.

Therefore, despite the increased asynchronous bandwidth and isochronous quality, the UWB system used in this implementation did not provide any advantages in terms of transmission distances. The system did however prove BToUWB functionality implemented by low-cost USB dongles under an open-source environment.

6. Future Work

Discovery of nearby devices and establishing a primary connection between two wireless devices were solely performed by the respective Bluetooth and UWB methods and subsystem. Utilizing the UWB physical layer for a hybrid approach to these tasks may provide even more advantages in terms of connection setup speed and device power saving. UWB connection setup and hibernation may be left to be completely handled by the UWB subsystem, or enabled under Bluetooth control upon a UWB service discovery. Power requirements and connection setup and wakeup speed will all be factors to consider upon investigating an optimal solution for these challenges.

The comprehensive scope of the project and lack of ample accessibility to UWB hardware and firmware limited the BToUWB analysis performed to the asynchronous bandwidth and isochronous quality with respect to transmission distances between two nodes with existing Bluetooth links. The system may be expanded to implement flow control for the proposed SCO over UWB system, implement UWB isochronous medium access for the transmission of SCO data, bypass the UWB WLP overhead with a direct UWB RC interface via the USB channel, or setup a multi-node network. Some connection setup tasks may also be performed by implementing UWB, including device discovery, synchronization or authentication.

7. Acknowledgement

The authors would like to thank Inaky Pérez-González and Ed Stock of Intel and the other developers of the Linux UWB driver for the debug support provided during this project implementation.

8. References

[1] Bluetooth Core Specification, Revisions 2.0., 2004. [Online]. Available: http://www.bluetooth.org

[2] Standard ECMA-368, “High Rate Ultra Wideband PHY and MAC Standard”, 2nd edition, ECMA International Standards, December 2007. [Online]. Available: http://www.ecma-international.org/publications/ standards/Ecma-368.htm

[3] Standard ECMA-369, “MAC-PHY Interface for ECMA-368”, 2nd edition, ECMA International Standards, December 2007. [Online]. Available: http://www.ecma-international.org/publications/standards/ Ecma-369.htm

[4] M. Hämäläinen, J. Saloranta, J. Mäkelä, I. Oppermann and T. Patana, “Ultra Wideband Signal Impact on IEEE802.11b and Bluetooth Performances”, IEEE Int. Symp. on Personal, Indoor and Mobile Radio Communications, PIMRC '03, pp. 2943 - 1947, 2003.

[5] A. Palin, J. Reunamäki, J. Salokannel and J. Ylänen, "Bluetooth Host Protocol Usage Over the Ultra Wideband Radio", IEEE Int. Symp. on Personal, Indoor and Mobile Radio Communications, PIMRC '06, pp. 1 - 4, September 2006.

[6] Open-Source SUSE Linux, Novell Inc, 2008: [Online]. Available: http://www.opensuse.org

[7] BlueZ Project, “Official Linux Bluetooth protocol stack”, BlueZ Website, November 2007. [Online]. Available: http://www.bluez.org

[8] J. Beutel and M. Krasnyanskly, “Bluetooth Protocol Stack for Linux - Linux BlueZ Howto”, Jeremy Thompson’s Personal Website, November 2001. [Online]. Available: http://www.jeremythompson.uklinux.net/RH8-0/bluezhowto.pdf

[9] E. van der Linde and G. P. Hancke, “An Investigation of Bluetooth Mergence with Ultra Wideband”, Third International Conference on Broadband Communications, Information Technology & Biomedical Applications, November 2008.

[10] “GUWA100U Wireless USB Host Adapter”, Online Product Description, IOGear Inc., http://www.iogear.com/product/GUWA100U, March 2008.

[11] Intel Corporation, “Intel Wireless UWB Link 1480 Ultra Wideband MAC”, Intel Online Website, http://www.intel.com/network/connectivity/products/uwb/index.htm, 13 January 2008.

[12] I. Pérez-González and R. Chatre, “Linux UWB + Wireless USB + WiNET”, Linux UWB Development, January 2008. [Online]. Available: http://www.linuxuwb.org/thewiki/Linux-UWB+WUSB+WiNET

[13] I. Pérez-González, “Linux UWB v1.8.9 Release Notes”, Linux UWB Online Community, http://www.linuxuwb.org/thewiki/README-1.8.9, 30 April 2007.

[14] Albert S. Huang, “The Use of Bluetooth in Linux and Location Aware Computing”, Master's thesis, Massachusetts Institute of Technology, Cambridge, 2005.

[15] E. van der Linde, “An investigation into the viability of UWB as lower-layer for Bluetooth”, University of Pretoria: Electroinic Thesis and Dissertations, http://upetd.up.ac.za/thesis/available/etd-11292009-191537, November 2009.